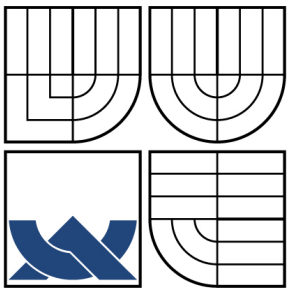
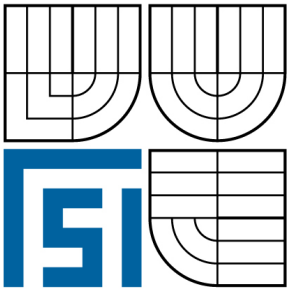


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA STROJNÍHO INŽENÝRSTVÍ

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

FACULTY OF MECHANICAL ENGINEERING

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

# POROVNÁNÍ SYSTÉMU OCTAVE A MATLAB NA ÚROVNI ZÁKLADNÍ MATEMATIKY

OCTAVE AND MATLAB COMPARISON IN TERMS OF BASIC MATHEMATICS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — RADEK ZAHRADNÍK

AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR KREJČÍ, Ph.D.

SUPERVISOR

BRNO 2008

# Zadání

(Z technických důvodů nebylo možné dodat zadání do termínu odevzdání)

# Abstrakt

Jak lidstvo stárne, tak se prohlubuje i poznání přírody, ve kterém žije. Bohužel na moderní problémy už lidský mozek jako takový nestačí a musíme si pomoci počítačem a moderními výpočetními metodami. Samotný počítač ovšem problém nevyřeší, k tomu je nutné adekvátní softwarové vybavení počítače. Cílem této práce je srozumitelně porovnat systémy Octave a Matlab z hlediska zaměnitelnosti v možných řešeních praktických problémů.

# Annotation

How is mankind grow older, its knowledge about nature, which in it lives, is getting deeper. Unfortunately human brain in itself became short for solving modern problems and we have to help ourselves with computers and modern computerised methods. But computers themselves can not solve the problems. We need to equip them with adequate software. Goal of this work is to compare Octave and Matlab systems in part of compatibility and in the solvings of practical problems.

# Prohlášení o autorství práce

Prohlašuji, že jsem zpracoval tuto bakalářskou práci pouze za pomoci vedoucího bakalářské práce a dané literatury.

V Brně dne 23. 5. 2008 .....

# OBSAH:

# 1. Úvod

Řešení moderních vědeckých problémů si dnes již nedovedeme představit bez použití výpočetní techniky. Samotný počítač ovšem nestačí. Je nutné ho vybavit adekvátním programovým vybavením a pracovníkem, který problém řádně zpracuje. Pokrok ve výkonu moderních počítačů se, díky multi-jádrovým CPU, stále drží Moorova zákona [2] a ceny výkonných multi-jádrových počítačů jsou nejnižší v historii. Ale tento výkon není vždy využit, protože v první řadě si žádá optimalizované aplikace pro více jader a schopnost pracovníka analyzovat problém tak, aby využil správně potenciál svého PC. Kromě náročnosti na lidský potenciál, cena takového softwaru je přiměřená práci, která je vynaložena při vývoji programového kódu a jeho optimalizaci. Zde je částečným řešením možnost použít tzn. svobodného software, který je zpravidla vyvíjen pod GNU GPL [3] licencí. Další výhoda toho řešení je ta, že použití tohoto systému si může dovolit kterékoliv školské zařízení včetně jednotlivce.

Primárně pro vědecké účely a numerická řešení byl vyvinut v roce 1954 imperativní programovací jazyk FORTRAN. I když tento programovací jazyk byl ve své době revoluční, s rozvojem doby a výpočetní techniky (*především nástup platformy PC*), která se stala dostupná pro širokou veřejnost, se FORTRAN stal spíše překážkou rozvoje. Tento důvod dal za vděk vzniku programovacích prostředí Octave a Matlab.

První prostředí Matlabu spatřilo světlo světa v roce 1970 a bylo navrženo jako snadný přístup studentům k softwarovým knihovnám LINPACK [4] a EISPACK [5], která byla napsána ve FORTRANu. První komerční prostředí Matlab, které již bylo napsáno v jazyce C, bylo uvedeno v roce 1984 jako produkt čerstvě založené firmy MathWorks. Projekt Octave začínal prakticky stejně. Jeho vývoj začal v roce 1988 jako studijní pomůcka pro výpočet chemických reaktorů a skutečný vývoj programu jako takového, začal v roce 1992 a první ostrá verze 1.0 byla uvedena v roce 1994 pod GNU GPL [3] licencí, pod níž je vyvíjen dodnes.

Cílem této práce je porovnat tyto dvě programová prostředí ne z hlediska toho, které je lepší (*vítěz by byl předem jasný*), ale z hlediska zaměnitelnosti těchto dvou programů v řešení praktických problémů, zaměnitelnosti kódu, skriptů a procedur a navést čtenáře jasnému rozhodnutí, zdali na zadaný problém mu bude stačit prostředí Octave nebo zdali bude nutné sáhnout po sofistikovanějším prostředí Matlabu.

Pro srovnání jsem si vybral aktuální verze obou programů, tj. Matlab R2007b a Octave 3.0.0 s grafickým uživatelským prostředím QtOctave 0.7.2. Samotné srovnání bude provedeno především v oblasti matematických operací, diferenciálních rovnic a možnosti rozšíření základních funkcí programu.

# 2. Popis funkcí

## 2.1. Základní matematické operace

### 2.1.1. Základní matematické operace Matlabu

Ze základních matematických operací jako sčítání, odčítání, násobení, dělení, mocniny a odmocniny reálných čísel, žádná z těchto funkcí v Matlabu nechybí. Dále nechybí ani logaritmy či trigonomické funkce a široké spektrum funkcí pro výpis maxima, resp. minima, zaokrouhlování čísel, návrat specifických hodnot atd.

### 2.1.2. Základní matematické operace Octave

Co se týče základních funkcí, má Octave stejnou funkčnost i zápis jako Matlab.

## 2.2. Práce s maticemi

### 2.2.1. Práce s maticemi v Matlabu

Název Matlab vznikl zkrácením anglických slov MATrix LABoratory (*laboratoř matic*), což přesně odráží skutečnost, v čem je Matlab nejlepší na světě. Matlab umožňuje práci s maticemi, jejichž složky mohou být jak reálná, tak komplexní čísla a také umožňuje práci s vektorovým polem, jehož prvky jsou různého datového typu. Dále Matlab podporuje řídké matice a specialitou Matlabu je úsporné ukládání toho typu matic. Není zapomenuto ani na sestavování matice matic či seskupování matic do bloků.

Práce se zápisem matic je jednoduchá a intuitivní. Od verze 6.5 je integrován nový JIT akcelerátor, který snižuje rozdíl ve výkonu oproti C++ a FORTRANu při výpočtech s maticemi za předpokladu vektorizovaného zápisu kódu. Matlab umožňuje všechny základní operace s maticemi jako je sčítání, odčítání, násobení, dělení zprava, dělení zleva, odmocniny a mocniny matic, determinant matice či transpozici matice. Dále umožňuje generování matic s náhodnými hodnotami, jednotkové matice, matice nul, resp. jedniček, výpisu horní, resp. dolní trojúhelní-

kové matice, tvorba Hilbertovi, Toeplitzovi, Vandermondeova, Hankelovi, Hadamardovi, doprovodné či diagonální matice. Do základních funkcí také patří funkce pro zjištění vlastní hodnoty matice, resp. singulárních hodnot dekompozice. Matlab rovněž umožňuje provádět skalární operace s prvky matic a do poněkud pokročilejších funkcí spadá řada funkcí na přeskupování matic, transpozici a testování hypotéz nad jednotlivými prvky.

### 2.2.2. Práce s maticemi v Octave

Systém Octave umožňuje také velmi pokročilou práci s maticemi. Složky matic mohou být také jak reálná, tak komplexní čísla. Octave umožňuje všechny základní operace s maticemi jako je sčítání, odčítání, násobení, dělení zprava, dělení zleva, odmocniny a mocniny matic, determinant matice či transpozici matice. Dále umožňuje generování matic s náhodnými hodnotami, jednotkové matice, matice nul, resp. jedniček, výpisu horní, resp. dolní trojúhelníkové matice, tvorba Hilbertovi, Toeplizovi, Sylvestrovi, Hankelovi, Vandermondeova, doprovodné či diagonální matice. Dále podporuje několik různých druhů funkcí pro tvorbu řídkých matic, kdy můžeme generovat řídké matice třeba pouze s jedničkami nebo náhodnými prvky. Do základních funkcí také patří funkce pro zjištění hodnosti matice, resp. singulárních hodnot dekompozice či rotace matice. Octave také umí operace po prvcích a taktéž má široké spektrum funkcí pro přeskupování matic.

## 2.3. Operace s komplexními čísli

### 2.3.1. Operace s komplexními čísli v Matlabu

Matlab umožňuje široké použití komplexních čísel při práci a to hlavě při simulaci signálů, jejich analýze atp. Zadání komplexního čísla umožňuje pomocí jednoduchých zápisů např. *3+3i* nebo *3+3j* a respektuje tím tak zažité konvence jak mezi elektroinženýry, tak inženýry obecně. Matlab umožňuje všechny základní početní operace s komplexními čísli, včetně jejich ukládání do vektorů, či matic. Mezi základní funkce patří funkce pro návrat imaginární hodnoty, testování imaginární části, komplexní konjugaci, převod reálného čísla na komplexní a

to jednak přímým převodem nebo složením dvou vektorů reálných čísel. Samozřejmostí je vykreslování komplexních čísel, viz kap. 2.9.1.

### 2.3.2. Operace s komplexními čísli v Octave

I Octave nerozlišuje mezi *3+3i* a *3+3j* a jako takový, taktéž nedělá rozdíly mezi operacemi s reálnými čísli a komplexními, tj. všechny výše zmíněné operace platí jak pro reálná čísla, tak pro komplexní. Taktéž umožňuje návrat imaginární hodnoty, komplexní konjugaci a vytvoření komplexního čísla složením dvou vektorů. Zápis funkcí je shodný s Matlabem.

## 2.4. Statistika

### 2.4.1. Statistika v Matlabu

V systému Matlab je statistika řešena komplexně pomocí externího dodatku, tzv. „toolboxu“ a to přímo od výrobce v základním instalačním balíčku. Co se týče základní statistiky typu střední hodnoty, medián, směrodatnou odchylku, rozptyl, kovarianci, korelaci, tak tyto funkce najdeme přímo v základním instalačním balíčku Matlabu a jejich zápis je opět stejný jako v Octave. Ovšem v toolboxu najdeme 23 různých testů hypotéz, grafické interaktivní nástroje pro znázornění 11 typů pravděpodobnostních rozdělení, generování 22 různých typů dat *(podle typu rozdělení pravděpodobnosti)*, analýzu dat v reálném čase, regresi dat, vykreslování histogramů, kvantilů, pravděpodobnostních rozdělení a celkově tento toolbox nabízí 424 různých statistických funkcí.

### 2.4.2. Statistika v Octave

Mezi základní statistické funkce Octave patří výpočet střední hodnoty *(harmonické, geometrické, či aritmetické)*, medián, směrodatnou odchylku *(vychýlenou i nevychýlenou)*, rozptyl, kovarianci, korelaci, exces a nesouměrnost. Ocenit zde musím funkci *statistics*, kdy po zavolání této funkce se nám vrátí vektor, jehož složky jsou hodnoty funkcí výše uvedené. Tuto funkce Matlab nemá. Dále Octave umí vypočítat p-moment sledované veličiny, vypočítat hodnoty binomického koeficien-

tu, permutaci, vrátit unikátní hodnoty ze souboru hodnot, setřídit ho či zjistit jeho rozsah. Octave také umožňuje testování 11 různých hypotéz či 22 typů pravděpodobnostního rozdělení. Samozřejmostí je i tvorba histogramů, kvantilů či vykreslování rozdělení pravděpodobnosti.

## 2.5. Prezentace výsledků

### 2.5.1. Grafický výstup v Matlabu

Prezentace výsledků v Matlabu je velmi propracováno a uživatelský komfort je na velmi vysoké úrovni. Je zde kladen vysoký důraz na názornost, rychlou orientaci v možnostech vykreslování a samotnou použitelnost výstupu. Celá škála příkladů v nápovědě spolu s výbornými interaktivními nástroji 2D výstupu názorně ukazují možnosti, které máme k dispozici. Uživatel se snadno zorientuje, v případě chyby ji snadněji odhalí, a může snadněji zvolit ideální formu výstupu.

Na rozdíl od Octave, 3D vykreslovací model Matlabu je mnohem lépe odladěný a je i lépe použitelný. I u 3D výstupu v Matlabu máme interaktivní nástroj, ve kterém můžeme přímo měnit, u námi zadané funkce, různé typy vykreslování spolu se stínováním či osami. Uživatel zde může názorně vidět rozdíly mezi jednotlivými typy vykreslování a opět si může zvolit ten nejvhodnější.

Matlab se v základních možnostech výstupu od Octave (*Gnuplotu*) vcelku neliší, liší se způsobem práce s ním. Taktéž máme k dispozici základní spojnicový graf plus speciální typy grafů, jako je sloupcovitý, schodovitý, histogram, errorbarový, čárový či polární graf. Rovněž může měnit barvu, typ křivky, znaky, kterými se křivka reprezentuje či měřítko os. Podstatnější však je, že tvorba kódu výstupu není tak náročná na přímé znalosti uživatele jak je tomu v Octave (*Gnuplotu*). Interaktivní nástroje, kterými Matlab disponuje, umožňují snadno měnit vstup či parametry výstupu a tyto změny se promítnou do kódu pod grafem, odkud si jej může uživatel zkopírovat a vložit kam potřebuje. V okně výstupu se nabízejí další možnosti jako snadný popis os, dat, vložení legendy nebo jiných entit do grafu, barevnou mapu nebo další osy, které můžeme využít, například když vykresluje více křivek do jednoho grafu s různými jednotkami. Velice jednouchá práce je i s regresí, kdy se na

vyžádání uživatele z nabídky zobrazí intuitivní panel s regresními křivkami, které se ihned vykreslují.
Pro pokročilejší uživatele je snadno přístupné i detailnější nastavení výstupu jako například barevný prostor, font použitý u popisu dat, jednotky, změnu defaultních hodnot tloušťky čar nebo velikosti grafu. Tyto nastavení je možno snadno exportovat/importovat a tak zjednodušit opakující se práci/úkony. Samozřejmostí je potom i velmi široké spektrum výstupního formátu, do kterých můžeme graf uložit a to např. JPG, TIFF, FIG (*formát umožňující i následnou editaci grafu*), AI, PCX, PCM a další.

### 2.5.2. Grafický výstup v Octave

Octave používá pro grafické výstupy nástroj *Gnuplot*. Ten umožňuje vykreslovat jak 2D grafy, tak 3D grafy. *Gnuplot* umožňuje širou škálu nastavení, od typu grafu *(viz níže)*, přes popisky dat, os až po nastavení tvaru grafu, měřítka nebo možnosti přizpůsobení grafu výstupnímu zařízení. Grafy mohou být proloženy křivkami, nebo reprezentovány body několika různých typů. Samozřejmostí je nastavení barev či vykreslování více křivek do jednoho grafu, standardně se při každém zavolání funkce plot, plátno překreslí. Je také možné vykreslit více grafů do jednoho okna výstupu. *Gnuplot* disponuje i několika speciálními grafy, jmenovitě sloupcový graf, histogram, polárním graf, grafy s logaritmickými měřítky *(na ose x, y nebo obou)*, errorbarový, čárový či schodovitý graf. Samotný *Gnuplot* rozeznává 40 matematický funkcí, které můžeme přímo použít při vykreslování. Jejich kompletní seznam, spolu s dalšími nematematickými funkcemi *Gnuplotu* můžeme najít v [6]. U výstupu můžeme také zobrazit na požádání mřížku pomocí funkce *grid* ta je ovšem pevně daná a uživatel ji nemá možnost dále ovlivňovat.
3D grafy mohou být, buď jako klasický výstup 3D nebo to mohou být síťové modely či jenom drátové modely. 3D výstup *Gnuplotu* bohužel není jeho silnou stránkou, při manipulaci s objekty se obraz značně trhá a to i při vypnutém anti-aliasingu.
Nápovědě Octave by prospělo, kdyby u sekce plotting bylo více obrázků ukazující, co která funkce znázorňuje, nemluvě o absenci interaktivi-

ty, která by usnadnila práci začátečníkům. Samotné okno výstupu *Gnuplotu* moc dalších možností, jak pracovat s vykreslenými daty, nemá. První možnost exportu je skrze napojení na schránku Windows, do které se výstup uloží v podobě čistě obrazových dat. Další možnosti exportu jsou založeny na vložení příkazu *print* do kódu a uživatel si může vybrat ze širokého spektra výstupních souborů, které může zpětně editovat.

# 3. Programování v systémech Matlab a Octave:

## 3.1. Programování v Matlabu

Programování a tvorba programů v Matlabu je velmi obsáhlé téma, které by vydalo na samostatnou bakalářskou práci, proto se zde omezíme pouze na stručné shrnutí klíčových vlastností. Jak již bylo řečeno v úvodu, Matlab je programové prostředí nebo také o něm můžeme říci, že je to programovací jazyk. Je to úplný programovací jazyk, čtvrté generace [13], tj. používá interpreter pro komunikaci s jádrem a uživatel tak nemusí definovat proměnné, ale pouze specifikovat co vlastně žádá. Matlab obsahuje všechny důležité příkazy pro tvorbu programů, jako jsou řídící podmíněné příkazy, větvicí příkazy, cykly a podobně. I přes svou jednoduchost umožňuje tvorbu složitých aplikací. O této skutečnosti na první pohled svědčí stovky toolboxů, které nejsou ničím jiným, než soubory dodatečných skriptů vytvořených nad rámec základní funkcionality samotného Matlabu.

### 3.1.1. Datové typy v Matlabu

Základním datovým typem je pole (*a to buď řídké nebo plné*), které se může dále „tvářit“ jako sada znaků, logická hodnota, JAVA objekt, buňka nebo jako čísla typu *integer*, *single*, *double*. Celkově lze v Matlabu použít 15 různých datových typů, jejich podrobný popis lze nalézt v nápovědě programu.

### 3.1.2. Tvorba programu v Matlabu

Základními stavebními prvky kódu jsou proměnné, funkce, klíčová slova, operátory (*logické, aritmetické a další*), výrazy, řídící příkazy a další. Z řídících příkazů máme k dispozici podmínkové (*if, switch*), regulační (*for, while*), poruchové (*try, catch*) a pomocné *break* a *continue*.
Tyto základní prvky sestavujeme dohromady do tzn. M-souborů (*mající příponu „.m“, proto M-soubor*). Tyto soubory mohou být dvojího typu, jeden, ve kterém se nacházejí pouze uživatelem definované funkce, první slovo v souboru musí být *function* a druhý – skriptovací, který může obsahovat libovolnou posloupnost příkazů Matlabu.
Nedílnou součástí programování jsou i „chyby.“ Při programování se mohou vyskytnou dva hlavní druhy chyb, chyba syntaxe a chyba při běhu programu. První druh odhalíme poměrně snadno, tj. Matlab nás při provádění příkazů na tyto chyby upozorní. Chyby při běhu programu se odstraňují mnohem obtížněji. Matlab nám tyto chyby pomáhá odstranit pomocí funkcí pro odlaďování. Mezi tyto funkce patří nastavení a odstranění bodu přerušení, výpis volání, změna kontextu lokálního pracovního prostoru a další.

### 3.1.3. Současný vývoj jazyka Matlabu

Celkově je dnešní vývoj Matlabu koncipován pro ještě lepší podporu programování, uživatelský komfort a široký záběr. Jazyk Matlabu je těsně integrován s jazykem JAVA a můžeme tak přímo použít řadu JAVA knihoven dostupných na internetu a vytvářet složité grafické rozhraní, známé třeba z prostředí Delphi. Dále můžeme použít i objekty z jazyku C a FORTRAN. Ve verzi R2008a Matlab plně podporuje již dokonce i objektově orientované programování. Tímto nechává Octave daleko za sebou a profiluje se zcela jinak.
Vývojáři z MathWorks věnovali poměrně dost pozornosti tomu, aby uživatel při tvorbě svého programu měl vše po ruce a měl snadný přehled o proměnných, historii zadání vstupů, správě složek. Kód můžeme strukturalizovat do elementárních částí – do buněk, se kterými můžeme nakládat jako s objekty. Pro snadné zvládnutí problematiky tvorby m-souborů, tvorby grafického prostředí či práce s prostředím připravili

vývojáři několik videokurzů, kde můžeme vše názorně vidět, popř. i vyzkoušet. Videokurzy můžeme nalézt v [15].

## 3.2. Programování v Octave

Octave je vyšší programovací jazyk, zaměřený především na vědeckou sféru, tudíž především vektorové operace a numerické výpočty. Klade si za cíl zjednodušit tyto operace, zpřístupnit je uživatelům, pro které není problém algoritmizovat, ale kteří nemají znalosti programování v C++ nebo FORTRANu, ze kterých syntaxově vychází a nechtějí zabřednout do úskalí, které direktivní programování s sebou přináší.

### 3.2.1. Proměnné v Octave

Octave používá místo překladače interpreter, který komunikuje s jádrem programu a není tedy nutno předem deklarovat datový typ proměnné.

Zabudované datové typy jsou především komplexní matice, které mohou být interpretovány jako i reálné a komplexní skaláry, řetězec znaků a datové struktury. Jméno proměnné musí být sekvence znaků, čísel nebo podtržítek a nemůže začínat číslicí. Octave nemá limit pro délku jmen proměnných. Ve jménech proměnných (stejně, jako v Matlabu) hraje roli velikost písma. Symboly *a* a *A* jsou rozdílné proměnné. Řetězce znaků v Octave se skládají ze sekvence znaků ohraničených dvojitými nebo jednoduchými uvozovkami. Vnitřně Octave ukládá řetězce jako matice znaků. Veškeré indexové operace tedy fungující pro matice fungují i pro řetězce.

### 3.2.2. Výrazy, cykly

Příkazy můžeme stavět pomocí výrazů a řídících cyklů. Výrazy jsou základní prvek tvrzení v Octave. Pomocí výrazu vypočítáme hodnotu a tu pak můžete vypsat, otestovat, uložit v proměnné, předat funkci nebo jí přiřadit novou hodnotu proměnné pomocí operátorů. Jako v jiných jazycích, výrazy v Octave obsahují proměnné, odkazy na pole, konstanty a volání funkcí stejně tak jako kombinace těchto rozličných operátorů.

Z cyklů máme k dispozici *if, for, switch, do-until, while, try* a *unwind_protect*, které jsou doplněny o pomocné příkazy *break* a *continue*. Řízení je provedeno pomocí příkazů *else, otherwise, end(název cyklu)*. Pomocí cyklů můžeme vytvořit taktéž rekurzi a je třeba mít na paměti, že v Octave je počet rekurzí omezen a řídí se přes systémovou proměnnou *max_recursion_depth*.

### 3.2.3. Práce s kódem

Zde platí totéž, co u Matlabu. Uživatel si svou syntaxi může uložit do skript souboru, který může jakoukoliv posloupnost příkazu Octave, kromě procedur vyžadující restart programu. A taky není třeba, aby uživatelem vytvořené funkce byly přímo v programu, ale mohou se uložit do m-souborů. Tyto soubory obsahují pouze deklarace funkcí a musí se nacházet v určeném adresáři, kde ho Octave může najít a načíst. Taktéž máme k dispozici nástroje pro odlaďování funkcí, které pracují obdobně jako v Matlabu a mnoha jiných jazycích. Osobně bych velmi doporučil použití GUI doplňku k Octave, které velmi usnadní manipulaci s kódem a jeho tvorbou obecně a snižují nároky na znalost direktivních operací typu SAVE a LOAD a umožňují se soustředit na kreativitu uživatele.

Do Octave je možno také importovat kód z jiného zdroje než je třeba Matlab. Pomocí funkce *mkoctfile* můžeme naimportovat kód, který byl napsán v C, C++ nebo ve FORTRANu.

## 3.3. Vzájemná zaměnitelnost kódu

Jak jsem již v úvodu napsal, Octave nevznikl jako konkurent Matlabu, dnes se však nejčastěji skloňuje jako freewareová alternativa Matlabu a nynější směr programu toto plně podporuje. I když se přidávají funkce z Matlabu, které chybí, je dbáno na kompatibilitu, velmi často se v nových verzích kromě oprav chyb objevují změny kódu a funkcionality, tak aby byl v souladu s Matlabem, nikdy nebude kompilace vytvořená v Matlabu 100% spustitelná v Octave. Spíše naopak, kód vytvořený v Matlabu bývá většinou kompatibilnější. Vývojářům jde především o

to, aby se uživatel nemusel učit nový programovací jazyk a přitom začlenit i některé užitečné prvky, které by se mohli hodit.

Míra kompatibility je přibližně 85%. Toto číslo je však orientační, stránky, které se zabývají vzájemnou kompatibilitou, trpí jedním společným neduhem a to je zastaralost vůči verzím a to především verzím Octave, který má dynamičtější vývoj. Jistou míru kompatibility navíc můžeme docílit při přidání startovacího parametru (*--traditional*) ke spouštěcímu souboru.

Typickým příkladem odlišnosti v zápisu syntaxe jsou například uvozovky, kdy Matlab podporuje pouze jednotné uvozovky ('matlab') a Octave používá jak ('octave') tak ("octave"). Jakmile se pokusíme spustit v Matlabu kód se dvojitými uvozovkami, tak zahlásí chybu. Co se týká M-souborů, Octave od verze 2.9.x dokáže načíst Matlab 7.x soubory a při použití nejnovější verze není potřeba zabývat se *file-version* managementem. Octave má stejný způsob definice funkce jako Matlab, tudíž každá, uživatelem definovaná funkce v Matlabu se v Octave nadefinuje stejně a zpravidla by měla i fungovat stejně pokud přímo v tělu funkce není použita vzájemná nekompatibilita.

Pro zlepšení kompatibility se obecně doporučuje specifikovat absolutní cesty, ale pouze pro SAVE a LOAD, pokud používáme zabudované funkce, ověřit si jejich přítomnost a názvy (*tato situace, kdy je funkce přítomna jak v Octave, tak v Matlabu, ale v Octave pod jiným názvem, nastává zřídka a bývá dosti často odstraněna v příští verzi, najdeme však záměrné výjimky, např. funkce "printf"*) a parametry, se kterými jsou funkce volány včetně jejich pořadí (*zde však taktéž platí výše zmíněné*).

Příklady zaměřené na vzájemnou kompatibilitu kódu, notoricky známé rozdíly, můžeme nalést v [7] nebo [12]. Obecně je velmi rozumné použít Google a uživatelská fóra Octave, např. [14].

# 4. Diferenciální rovnice

Abychom mohli kvalitativně srovnat Matlab a Octave v oblasti řešení diferenciálních rovnic (*dále jen DR*) musíme si specifikovat několik základních poznatků z teorie numerického řešení diferenciálních rovnic.

Nebudeme zacházet do přílišných podrobností a rigorózních matematických postupů, jelikož problematika řešení DR není předmětem této práce, vše zde uvedené lze podrobně nalézt například v [10].

# 4.1. Základní poznatky numerického řešení DR

## 4.1.1. Definice DR

Obyčejná diferenciální rovnice je rovnice, jejíž neznámá je funkce (*jedné proměnné*), a která obsahuje derivaci (*příp. derivace*) této neznámé funkce.

DR bývá nejčastěji specifikována těmito ekvivalentními předpisy:

$$y'(t) = f(t, y), \qquad \frac{dy}{dt} = f(t, y)$$

přičemž *y* je taktéž funkcí *t*, tj. *y (t) a tato* funkce je řešením, které hledáme.

## 4.1.2. Numerické řešení DR

Funkci $f(t, y)$ někdy nazýváme stavová rovnice a v praxi bývá obtížné tuto rovnici analyticky řešit. Pak tuto rovnici řešíme numericky pomocí diskrétních časových kroků $\Delta t$:

$y(t + \Delta t) = y(t) + D(t, y)\Delta t$

kde $D(t, y)\Delta t$ je směrová funkce, která se snaží aproximovat $y'(t)$ co možná nejpřesněji.

## 4.1.3. Druhy problémů

Obecně můžeme rozdělit problémy, které se snažíme popsat DR na:

- **Tuhé** (*stiffness*)
    - Tento pojem (*problém*) dosud nemá přirozenou matematickou definici. Jeho popis se prování pomocí několika tezí, některé z nich nalezneme v bodu 4.1.5.
- **Standardní** (*non-stiffness*)

- Tyto problémy jsou doplňkem tuhých problémů, tj. všechny ostatní.

### 4.1.4. Druhy numerických řešení DR

Dnes již známe mnoho numerických metod pro výpočet, ale všechny mají jedno společné, jedná se o diskretizaci spojitého problému.

- **Vícekrokové metody explicitní**
  - U těchto metod se k výpočtu nové funkční hodnoty použije více než jedna (*předchozí*) hodnota stavové rovnice a funkce. Řád metody je určen z počtu předchozích kroků použitých při výpočtu.
- **Více krokové metody implicitní**
  - U těchto metod se při výpočtu nové funkční hodnoty používá více než jedna hodnota stavové rovnice a funkce a to včetně hodnoty příslušející právě zjišťované. Symbolicky naznačeno: $y_{n+1} = y_n + a \cdot (f_{n+1} + f_n)$
- **Metody podle Rudge-Kutta**
  - Často vyskytující se metody, které jsou dosti přesné. Mohou být jak explicitní tak implicitní. Řád metody určuje interpolační polynom. Nejčastěji se používá metoda 4. řádu, tzn. RK4. Do této skupinu můžeme zařadit speciální případ metody 1. řádu – Eulerovu metodu, která byla první metodou pro řešení DR.
- **Metody Prediktor-Korektor**
  - Tyto metody vnikly spojením explicitních a implicitních metody, kdy funkční hodnoty odhadneme explicitní metodou a zpřesníme pomocí implicitní. Tyto metody se vyznačují malou oblastní A-stability a jsou tudíž nevhodné pro řešení tuhých řešení.

### 4.1.5. Co je to tuhost?

Tuhé problémy se vyznačují především tím, že pokud na ně aplikujeme jinak velmi efektivní numerické metody např. explicitní metody dle Runge-Kutta, bude výpočet jako takový zcela neefektivní či přímo nepraktický. Pro pevnou délku kroku dostáváme numerické řešení, které velmi často osciluje a vždy explozivně roste. Při automatickém řízení kroku se dostáváme na velmi malé hodnoty kroku a objem výpočtu neúměrně roste.

V praxi bylo zjištěno, že méně přesnější metody, u kterých se krok řídí velikostí lokální chyby, jsou pří řešení tuhých problémů velmi efektivní. Tento poznatek shrnuje teze: *Tuhost soustavy ODR se projevuje tím, že při jejím numerickém řešení délku kroku omezuje spíše stabilita metody než její přesnost.*

Toto se v praxi popisuje pomocí A-stability a obecně můžeme říci, že čím má daná metoda větší A-stabilitu, tím je pro řešení tuhých problémů vhodnější. Tuto vlastnost obecně splňují speciální implicitní vícekrokové metody, označované jako metody zpětného derivování nebo také zkratkou *BDF*, která vznikla z anglického originálu „Backward Differentiation Formula“. K posouzení, zdali problém je či není tuhý, slouží praktické kritérium:

*Jestliže numerická metoda s omezenou oblastí A-stability, aplikována na počáteční problém, je nucena v jistém intervalu integrace používat krok, jehož délka je nepřiměřeně malá vzhledem k hladkosti přesného řešení v tomto intervalu, pak to znamená, že problém je v tomto intervalu tuhý.*

## 4.2. Diferenciální rovnice v Matlabu

Vývojáři z MathWorksu věnovali problematice DR velkou pozornost a Matlab překonává Octave v problematice výpočtu DR snad ve všech směrech. Můžeme řešit algebraické DR (*zkráceně DAR*) s počáteční podmínkou, obyčejné DR (*zkráceně ODR*) s počáteční podmínkou i okrajovou podmínkou, parciální DR a DR se zpožděním s počáteční podmínkou. Pro ODR a DAR s počáteční podmínkou je k dispozici 8 řešitelů + pomocné funkce pro prezentaci výsledků, práci s nastavením parametrů řešitele, vyčíslování a prodloužení intervalu výsledků. K ře-

šení ODR s okrajovou podmínkou máme k dispozici jednoho řešitele, podpůrné funkce pro odhad počáteční hodnoty, vyčíslení výsledků, správu parametrů řešitele a 6 názorných příkladů. Pro parciální DR máme k dispozici taktéž jednoho řešitele s 5 interaktivními funkcemi, které nám ilustrují výpočet v praxi. U diferenciálních rovnic se zpožděním máme k dispozici 2 řešitele, opět podpůrné funkce pro vyčíslení výsledků, správu parametrů řešitele a 3 názorné příklady. Nápověda je velmi propracovaná, přehledná a najdeme zde vše, co bychom mohli potřebovat.

### 4.2.1. Netuhé ODR a DAR s počáteční podmínkou

- **ode45**
    - Tato metoda je založena na explicitní Runge-Kuttově formuli a Dormand-Princově páru. Je to jednokroková metoda vhodná pro "první pokus."
- **ode23**
    - Tato metoda je založena na explicitní Runge-Kuttově formuli, Bogackiho a Shampineho páru. Je to taktéž jednokroková metoda, snese mírnou tuhost a při nižší toleranci je efektivnější než *ode45*.
- **ode123**
    - Vícekroková metoda Prediktor-Korektor založena na Adams-Bashforth-Moultonově spojené formuli s proměnlivým řádem formule. Při přísných nárocích na přesnost, může být více efektivní, než *ode45*.

### 4.2.2. Tuhé ODR a DAR s počáteční podmínkou

- **ode15s**
    - Vícekroková metoda s proměnlivým řádem, která je založena na numerických diferenčních formulích, volitelně můžeme použít formuli zpětného derivování. Je vhodná pro případy,

kde tuhost předpokládáme nebo o ní víme a také tam, kde je *ode45* selhal nebo se ukázal značně neefektivní.

- **ode23s**
  - Tato metoda je založena na modifikované Rosenbrockově formuli 2. řádu. Jedná se o jednokrokovou metodu a při nižších tolerancích může být více efektivní než *ode15s*. Může s úspěchem vyřešit problémy, u kterých se použití *ode15s* ukázalo neefektivní.

- **ode23t**
  - Implementace lichoběžníkového metody s použitím volného interpolantu, vhodné především pro mírně tuhé problémy a potřebujeme-li řešení bez numerických tlumení.

- **ode23t**
  - Implementace *TR-BDF2* metody, implicitní Runge-Kuttovi formule o dvou stupních, kde první stupeň je lichoběžníková formule a druhá je formule zpětné derivace 2. řádu. Stejně jako *ode23s*, může být tento řešitel více efektivní než *ode15s* při nižší toleranci.

### 4.2.3. Diferenciální rovnice se zpožděním

- **dde23**
  - Tento řešitel kalkuluje DR s konstantní délkou zpoždění. Při zpoždění menším než je velikost kroku, je použita explicitní Runge-Kuttova formule 2. nebo 3. řádu, při zpoždění větším, než velikost kroku, je použita implicitní Runge-Kuttova formule, vypočtená metodou Prediktor-Korektor.

- **ddest**
  - Tento řešitel kalkuluje DR s obecnou délkou zpoždění. Používá explicitní Runge-Kuttova formuli 4. řádu, přičemž používá iterace tak, aby kroky byly větší než zpoždění.

## 4.3. Diferenciální rovnice v Octave

Octave disponuje dvěma funkcemi pro řešení diferenciálních rovnic. Obě jsou založeny na již ověřených ODE [8] řešitelích napsaných ve FORTRANu. Důležité je podotknout, že v Octave můžeme řešit ODR a DAR.

### 4.3.1. Lsolve – Hindmarshovův ODR řešitel

Tento řešitel [9] byl napsán A. C. Hindmarshem v první polovině 90. let, je nezávislý na platformě a jeho implementace v Octave je určena pro běžné obyčejné diferenciální rovnice (*dále jen ODR*) ve tvaru:

$$\frac{dy}{dt} = f(y,t)$$

s počáteční podmínkou pro $t = 0$. Tato funkce nám vrátí řešení ve tvaru matice, kde každý řádek odpovídá prvku vektoru *t*. Při volání *lsolve* je samozřejmě nutné zadat argument s řetězcem, ve kterém odkazujeme na funkci, která má být využita k řešení. Tato funkce musí být ve tvaru:

$$xdot = f(x,t)$$

kde *x* jsou vektory a *t* je skalár. Pokud výše uvedený řetězec je dvou prvkový, tak druhý argument se použije při výpočtu Jakobiánu. Funkce pro Jakobián musí mít tvar:

$$jac = j(x,t)$$

kde *jac* je matice parciálních derivací.

Řešitel *Lsolve* má zabudováno několik mechanismů pro ověření stavu kalkulace a jeho úspěšnosti. Jeho implementace v Octace dle oficiálních pramenů má pouze nástroj *istate*, tj. parametr, který nám po úspěšném výpočtu vypíše kontrolní hodnotu, signalizující úspěch a v případě neúspěchu, vrátí chybovou zprávu. Bohužel jak přímo tento parametr v Octave funguje, se mi nepodařilo zjistit, jelikož oficiální dokumentace je velmi strohá a neadekvátní problému řešení ODR (*a opět chybí názorné příklady*). Odkazuje se na manuál *Lsolve* řešitele, ale ten jako takový popisuje, co tento parametr znamená a jakých hodnot může nabývat, ale přímo v tomto manuálu není ani slovo o tom, jak je tato funkce zabudována v Octave (*tento manuál vznikl před vznikem Octave*). A ne-

oficiální prameny odkazují, když ne jeden na druhý, tak na oficiální nápovědu, zmíněnou výše.
Podrobný popis struktury a metody použití Lsolve řešitele a obecně problematiky numerického řešení ODR s tím spojené, můžeme nalézt v [9].

### 4.3.2. Možnosti nastavení Lsolve řešitele

Mezi možnosti nastavení *Lsolve* řešitele patří absolutní a relativní tolerance, maximum, minimum kroku včetně limit kroků. Samozřejmostí je výběr integrační metody, kdy máme na výběr z:

- **Adamsovy metody** (*adams*)
    - Jedná se s největší pravděpodobností o spojení Adams-Bashorfhovi metody, která je explicitní, s metodou Adams-Moultonovy metody, která je implicitní. Spolu tvoří metodu prediktor-korektor. Tato metoda nahrazuje hodnotu stavové funkce interpolačním polynomem, který je vyjádřen zpětnými diferencemi a jeho řád, určuje i řád metody. Tento polynom je vhodný pro sledování lokální chyby a lze ho tedy snadno použít pro řízení délky kroku. Ovšem tato metoda se nehodí pro řešení tuhých problémů.
- **Metoda zpětného derivování** (*bdf*)
    - Tato metoda se vyznačuje hlavně tím, že délka kroku je řízena především stabilitou metody a ne velikostí chyby, je tudíž vhodná pro řešení tuhých problémů. Je to dáno velkou oblastí A-stability, která je dokonce pro tuto metodu 1. a 2. řádu nekonečně velká. Nejvyšší řád metody je 6, kdy při tomto řádu je vždy konvergentní.
- **non-stiff**
    - U této metody není použit Jakobián, i když je k dispozici.
- **stiff**

- U této metody je použita taktéž BDF metoda s tím, že ji nevadí, pokud není k dispozici Jakobián. V tom případě Lsolve řešitel vypočítá diferenciální aproximaci Jakobiánovy matice.

### 4.3.3. Daspk/Dasrp – Petzoldovův DAR řešitel

V matematice je diferenciální algebraická rovnice obecnou formou DR, dána v implicitní formě. Mohou být zapsány takto:

$$f[y'(t), y(t), t] = 0 \qquad f\left[\frac{dy}{dt}, y(t), t\right] = 0$$

s počátečními podmínkami

$$x(t = 0) = x(0), \qquad xdot(t = 0) = xdot(0)$$

*Daspk* řešitel je určen pro samostatnou rovnici a *Dasrp* řešitel pro soustavu rovnic. Po zavolání nám vrátí řešení ve tvaru matice, kde každý řádek odpovídá prvku vektoru *t*. Při volání *Daspk i Dasrp* je samozřejmě nutné zadat argument s řetězcem, ve kterém odkazujeme na funkci, která má být využita k řešení. Tato funkce musí být ve tvaru:

$$res = f(x, xdot, t)$$

kde *x*, *xdot*, *res* jsou vektory a *t* je skalár. Pokud výše uvedený řetězec je dvou prvkový, tak druhý argument se použije při výpočtu modifikovaného Jakobiánu. Funkce pro Jakobián musí mít tvar:

$$jac = j(x, xdot, t, c)$$

kde *jac* je matice parciálních derivací.

I v tomto řešiteli je přítomen nástroj *istate*, který po úspěšném výpočtu bude větší než nula.

### 4.3.4. Možnosti nastavení Daspk/Dasrp řešitele

U Daspk řešitele můžeme nastavovat opět absolutní a relativní přesnost, maximum, minimum a limit kroku. Dále můžeme nastavovat počáteční podmínku konzistence výpočtu, heuristiku startovací podmínky, algebraické proměnné, typ jejich nesouměrnosti.

U *Dasrp* řešitele můžeme taktéž nastavovat absolutní i relativní toleranci, maximum, minimum a limit kroků.

# 5. Stavové systémy

## 5.1. Popis stavových systémů

V teorii řízení, stavovým systémem modelujeme reálný fyzikální systém se vstupem a výstupem. Nejobecnější popis stavového systému je následující:

$$\dot{x}(t) = A(t)x(t) + B(t)u(t)$$
$$y(t) = C(t)x(t) + D(t)u(t)$$

Kde *x* je stavový vektor, *y* je vektor výstupu, *u* je vektor vstupu, *A* je stavová matice, *B* je matice vstupu, *C* je matice výstupu a *D* je matice dalších vlivů.

Model studujeme buď pomocí vlastních hodnot matice *A* nebo jednodušeji pomocí přenosu, což je obraz výstupní veličiny ke vstupní při nulových podmínkách. Přenos může být vyjádřen pomocí koeficientů diferenciální rovnice, pomocí nul a pólů nebo časových konstant. Na tomto modelu potom modelujeme, resp. testujeme chování systému, zpravidla odezvou na jednotkový skok, přechodovou funkci nebo Diracův signál a impulzní funkci.

## 5.2. Zápis stavových systémů

### 5.2.1. Zápis stavových systémů v Matlabu

U Matlabu máme k dispozici tyto funkce: Pro systém definovaný pomocí přenosu je to *tf*, pro systém zapsaný pomocí matic máme *ss* a pro systém popsaný pomocí nul a pólů slouží funkce *zpk*. Tyto 3 funkce mohou definovat modely spojité i diskrétní a taktéž s nimi můžeme vytvořit modely s více vstupy a výstupy. Dále můžeme zapsat systém pomocí dat z frekvenční odezvy systému. K tomu nám slouží funkce *frd*, která definuje pouze spojité systémy a taktéž může vytvořit systém s více vstupy a výstupy. Takto vytvořené systémy můžeme otestovat na stabilitu, aplikovat funkce *step* nebo *impulse* nebo převádět popisy systému mezi sebou stylem stavový popis na přenos = *ss2tf*.

### 5.2.2. Zápis stavových systémů v Octave

V Octave můžeme zapsat systém pomocí několika funkcí a záleží jenom na nás, jak systém nadefinujeme. Pro systémy s konečným impul-

zem odezvy máme funkci *fir2sys*, pro využití rovnice přenosu máme funkci *tf*, pro stavový popis pomocí matic máme funkci *ss* a pro popis pomocí pólů a nul máme funkce *zp*. Samozřejmě tyto popisy je mezi sebou možno převádět stylem přechodová funkce na stavový popis = *tf2ss*. Analogicky můžeme odvodit další. Takto popsané systémy můžeme testovat. Pro odezvu na jednotkový skok je funkce *step*. Pro odezvu na Diracův signál je k dispozici funkce *impulse*. Aplikace těchto funkcí je jednoduchá, kdy ji zavoláme s názvem našeho systému. Výše zmíněné funkce slouží k definici diskrétních systémů a to tak, že k nim přidáme navíc jeden parametr, který se interpretuje jako vzorkovací perioda.

# 6. Specifika systému Octave

## 6.1. Symbolická matematika v Octave

Symbolická matematika, či symbolická manipulace se vyznačuje tím, že se elementy výpočtu nevyhodnocují, avšak lze s nimi provádět transformace, zjednodušení nebo i další operace. V základní verzi Octave symbolickou matematiku nenajdeme, můžeme ji však doinstalovat pomocí doplňkového balíčku, stejně jako u Matlabu. V Matlabu je však tento toolbox přítomen v hlavním instalačním balíčku a je snadno na dosah. Balíček pro Octave je součástí projektu Octave Forge viz 7.2.1 a můžeme ho nalézt v [11]. Je založen na GiNaC projektu, podporuje především goniometrické a exponenciální funkce a autor sám uvádí, že největší nedostatek je absence podpory pro symbolické matice. Kompletní specifikace a dokumentaci můžeme taktéž nalézt v [11].

## 6.2. Doplňky pro Octave

Octave jako takový podporuje instalaci dalších balíčků. V Matlabu jim říkáme toolboxy a pro Octave se vžil název „*packages*.“ Pokud začneme hledat doplňkové balíčky na internetu, skončíme zcela jistě na stránkách projektu Octave Forge, vše o něm a o jeho významnosti se můžeme dočíst v bodě 7.2.1.

Mezi další důležité doplňky patří zcela jistě GUI doplňky, tedy grafické uživatelské prostředí, které zlepšuje komunikaci uživatele s jádrem programu. Velmi výkonný je například zde použitý *QtoOctave*, který se svým vzhledem a funkčností přibližuje rozhraní Matlabu.
Další doplňky, které můžeme najít, jsou grafické nadstavby *Octaviz* a *Octplot*, které realizují komplexní vizualizační systémy. Ovšem tyto nadstavy jsou zkompilovány pouze pro Linux a nemohl jsem je bohužel vyzkoušet.

### 6.2.1. Octave Forge

Častým problémem "svobodných" projektů, které se tvoří pod GNU/GPL licencí, je decentralizovatelnost, kdy více lidí pracuje na jedné a té samé věci nebo ji navzájem přepracovávají.
Otevřené prostředí Octave je rozené pro rozšíření ze strany uživatelů a Octave Forge je projekt, který se snaží centralizovat vývoj instalačních balíčků pro Octave. Dnes tento projekt zaštituje 47 hlavních rozšíření pro (*pouze namátkou*) audio, video, zpracování signálu, kombinatoriku, databáze, finanční matematiku, teorii informace, lineární algebru, fyzikální konstanty, vykreslování, statistiku, řetězce, analýzu nepravidelného vzorkování a další. Dále máme k dispozici 23 extra doplňků pro podporu řešení rovnic vyskytujících se ve stavebním inženýrství, objekty COM a funkcionalitu ve Windows, vícerozměrné integrace, podporu JAVA rozhraní, alternativní grafické nástroje, akceleraci výpočtu na multi-jádrových procesorech a další.
Pro ty, kteří to myslí s Octave opravdu vážně, je Octave Forge absolutní nutnost.

# Závěr

Octave i Matlab jsou velmi mocné nástroje pro vědecké výpočty a komplexnost obou systémů je daleko složitější a obsáhlejší, než mohu v této práci popsat. Je známo, že placený software bývá zpravidla lepší, než jeho Open-source protějšek. Nejinak je tomu i v případě Octave a Matlabu. Komplexnost řešení, které poskytuje Matlab po téměř 40 letech *(z toho 24 let komerčního)* vývoje nepřekoná ani sebevětší základna uživatelů tvořící pod GNU/GPL licencí. Ovšem pro systém Octave hovoří dvě silné výhody a to, že již zmíněná GNU/GPL licence a i když tak nebyl původně zamýšlen, shoda s Matlabem v zápisu základních i mnoha pokročilejších funkcí. Hlavním cílem je to, aby se uživatel při přechodu z Octave na Matlab nemusel učit nový programovací jazyk, což se vývojářům Octave daří. Navíc jako jeden z mála pokročilejších softwarů pro numerické výpočty, je zcela zdarma a to včetně doplňků k základní instalaci. Je tak především vhodný pro zastánce otevřeného softwaru, začínající programátory, studenty napříč všemi vědními obory a jedince, kteří mají potřeby numerických výpočtů.
Může také najít své uplatnění v komerční sféře. A to především u firem s menším rozpočtem nebo u firem, které musí čas od času řešit problém, který přímo nesouvisí s jejich podnikatelským záměrem, třeba např. statistická analýza dat, problémy automatizační techniky nebo třeba z čistě prezentačních důvodů. Lze ho také za jistých předpokladů použít spouštění m-souborů původně vytvořených v Matlabu a tak není nutné kupovat jeho licenci. Kde však Octave může najít využití je vědecká sféra. Díky otevřenosti kódu je flexibilnější a může se snadněji přizpůsobit na míru řešenému problému, což uzavřená licence komerčních programů neumožňuje.
Na druhou stranu, v základní verzi (*okno příkazové řádky*), je Octave dosti těžkopádný a skoro nepoužitelný v praxi. A asi největším problémem Octave je jeho nápověda. Obsahově často neobsahuje podrobnosti k funkcím, chybí hlubší problematika popisujících problémů a místy je dosti nepřehledná. Z technického hlediska je bohužel k dispozici pouze fulltextové vyhledávání a chybí interaktivita vzájemných odkazů, která by velmi ulehčila a zrychlila práci.
V případě Matlabu je situace vcelku jasná. Je ve svém oboru jedničkou na trhu a dokáže řešit problémy z několika vědeckých disciplín, je orientován na objemné a složité výpočty, popř. na krátké výpočetní časy. Má propracovanou nápovědu, snadno se v něm orientuje a cena je úměrná jeho schopnostem.
Pokud bych si měl vybrat prostředí pro studium, výpočty a případně i praxi, byl by to Octave, ale zcela vážně bych uvažoval o studentské licenci Matlabu a záleželo by pouze na její ceně a omezeních

# Literatura


**[1] OCTAVE – ČESKÝ PRŮVODCE PROGRAMEM;** http://www.octave.cz/**, 2006-6**

**[2] MOORŮV ZÁKON;** http://cs.wikipedia.org/wiki/Mooreův_zákon**, 2005**

**[3] GNU GPL;** http://cs.wikipedia.org/wiki/GNU_General_Public_License**, 2008**

**[4] LINPACK;** http://en.wikipedia.org/wiki/LINPACK**, 2008**

**[5] EISPACK;** http://en.wikipedia.org/wiki/EISPACK**, 2008**

**[6] GNUPLOT MANUÁL;** http://www.gnuplot.info/docs/gnuplot.html**, 2008**

**[7] VZÁJEMNÁ KOMPATIBILITA KÓDU;**
http://wiki.octave.org/wiki.pl?MatlabOctaveCompatibility**, 2008**

**[8]** ODE – Open dynamics engine**;** http://www.ode.org/**, 2008**

**[9] LSOLVE – LIVERMORE SOLVER FOR ORDINARY DIFFERENTIAL EQUATIONS;**
https://computation.llnl.gov/casc/nsde/pubs/u113855.pdf**, 1993**

**[10] ČERMÁK, L., NUMERICKÉ METODY II, SKRIPTUM VUT V BRNĚ, CERM, 2004**

**[11]** Octave Forge; http://octave.sourceforge.net/**, 2008**

**[12] OCTAVE <--> MATLAB COMPATIBILITY DATABASE;**
http://users.powernet.co.uk/kienzle/octave/matcompat/**, 2001-2**

**[13] VÝVOJ PROGRAMOVÁNÍ A PROGRAMOVACÍCH JAZYKŮ;**
http://www.fi.muni.cz/usr/jkucera/pv109/2002/xkriz1.htm**, 2002**

**[14] NABBLE – OCTAVE FÓRUM;** http://www.nabble.com/Octave-f1895.html**, 2008**

**[15] VIDEO KURZ PRÁCE S MATLABEM;**
http://www.mathworks.com/products/matlab/demo.jsp**, 2008**